

Meshfree C^2 -Weighting for Shape Deformation

Chuhua Xian^{1,2} Shuo Jin¹ Charlie C. L. Wang¹

¹The Chinese University of Hong Kong ²South China University of Technology

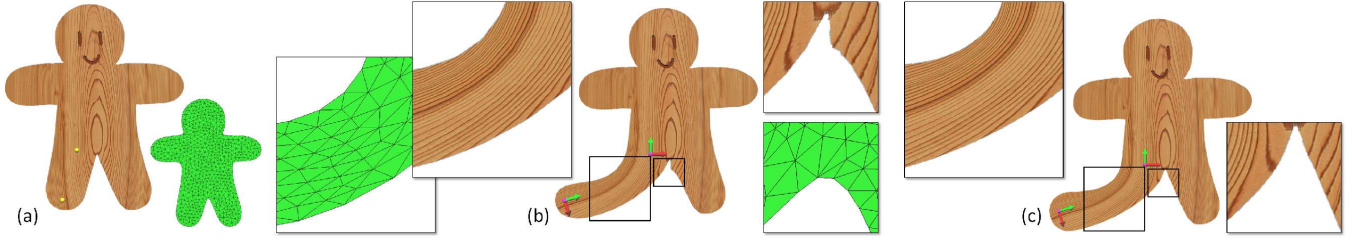


Figure 1: Deformation using meshfree C^2 -weighting. (a) Handle-driven deformation based on linear blending is an intuitive method for the interactive shape manipulation. (b) Artifacts caused by highly distorted triangles can be generated from the weights computed on an originally well-meshed domain. (c) We propose a meshfree framework to generate C^2 -continuous weights for linear blending based deformation. Our approach inherits the merits of mesh-dependent weighting schemes meanwhile bringing the weighting method to the resolution of infinity.

Abstract

Handle-driven deformation based on linear blending is widely used in many applications because of its merits in intuitiveness, efficiency and easiness of implementation. We provide a meshfree method to compute the smooth weights of linear blending for shape deformation. The C^2 -continuity of weighting is guaranteed by the carefully formulated basis functions, with which the computation of weights is in a closed-form. Criteria to ensure the quality of deformation are preserved by the basis functions after decomposing the shape domain according to the Voronoi diagram of handles. The cost of inserting a new handle is only the time to evaluate the distances from the new handle to all sample points in the space of deformation. Moreover, a virtual handle insertion algorithm has been developed to allow users freely placing handles while preserving the criteria on weights. Experimental examples for real-time 2D/3D deformations are shown to demonstrate the effectiveness of this method.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: shape deformation, meshfree, closed-form formulation, linear blend skinning

1 Introduction

Shape deformation techniques have various applications in computer graphics for image manipulation, geometric modeling and animation. Compared with other deformation strategies, handle-driven methods outperform others as they are intuitive, effective and easy-to-implement in many different scenarios. Using handles, users can bind a shape Ω with the handles and then manipulate their

locations and orientations to drive the deformation of Ω . Specifically, each handle H_i with $i = 1, \dots, m$ is defined as a local frame with its origin $\mathbf{h}_i \in \Omega$. After defining an affine transformation \mathbf{T}_i for each handle H_i , the deformation of Ω is realized by computing the new position of each point $\mathbf{p} \in \Omega$ via a linear blending of affine transformations $\mathbf{T}_i \mathbf{p}$. The linear blending is weighted by fields $w_i : \Omega \mapsto \mathbb{R}$ associated with handles H_i . Basically, to achieve an intuitive and high-quality deformation, the following criteria on the weights are demanded: smoothness, non-negativity, partition-of-unity, locality/sparsity, and no-local-maxima (see the analysis given in [Jacobson et al. 2011]).

The recent advancement of technology focuses on computing weights of blending on a discrete form of domain (i.e., meshes are employed to determine piecewise linear fields of weights). Weights are computed on the mesh nodes via minimizing some discrete differential energies (e.g., biharmonic, triharmonic and quatrharmonic used in [Jacobson et al. 2012b]). After incorporating the hard constraints according to the criteria on weights, the weights are determined on mesh nodes with the help of non-linear optimization. However, this is time-consuming. As a result, the insertion of new handles cannot be realized in real-time as new routines of non-linear optimization need to be taken. Moreover, the determined weights are mesh-dependent. For a symmetric shape to be deformed that is asymmetrically meshed, the computed weights for a handle located at the symmetric positions can rarely be symmetric. For poorly meshed computational domains, the artificial distortion caused by the elements of poor shape is more serious (as illustrated in Fig.1). Although the artifacts can be reduced by increasing the density of meshes, this will further slow down the computation. Ideally, the distribution of weights should only be affected by the shape to be deformed and the locations of handles, which indicates mesh-independence. Existing mesh-independent approaches in literature for handle-driven deformation (e.g., [Singh and Fiume 1998; Milliron et al. 2002; von Funck et al. 2006; Sumner et al. 2007]) can only satisfy subsets of the demanded properties on weights. This motivates our work on investigating a new meshfree method to determine weights for shape deformation.

In this paper, we formulate the evaluation of weights in a closed-form so that the deformation framework based on this gains the benefit of flexibility – i.e., the response of inserting new handles is real-time. Specifically, the time cost of inserting a new handle is linear to the number of samples used to represent the domain of

computation. The basis function formulated in this approach can guarantee the properties of smoothness, non-negativity, partition-of-unity, locality/sparsity, and no-local-maxima, all of which are necessary to ensure a deformation of high-quality.

The main results of our work are as follows:

- We present a meshfree method to determine linear blending weights with C^2 -continuity for real-time deformation. The weights are formulated in a closed-form of basis functions centered at the handles (details are given in Section 3.1). After decomposing the region to be deformed by the Voronoi diagram of handles, aforementioned criteria of shape deformation are all ensured (see the analysis in Section 3.2).
- A virtual handle insertion algorithm is proposed in Section 4 to guarantee the locality and sparsity of weighting so that a deformation interpolates the transformations defined on handles. The virtual handles are added to let the supporting region of the basis function defined on a handle not cover the origins of any other handles (see the algorithm in Section 4.1).
- After constructing the Voronoi diagram of all handles (including user-input and virtual ones), its dual-graph gives a connectivity of the handles. We compute harmonic fields on the graph to determine the transformations of virtual handles according to the transformations specified on the user-input handles (see Section 4.2). It is found that the transformations determined in this way lead to a shape-aware deformation following the intention of user input.

With the help of a discrete implementation on point samples introduced in Section 5, an efficient and effective meshfree approach has been developed for handle-driven shape deformation. 2D/3D experimental results are shown in Section 6 to demonstrate the performance of our approach.

2 Related Work

Shape deformation is an important research area in image manipulation and geometric modeling. There are a large amount of existing approaches in literature. The purpose of this section is not for a comprehensive review. We only focus on discussing the handle-driven deformation approaches.

Mesh-based techniques for discrete geometry modeling and processing have been widely explored in the past decade. Typical approaches including variational surface deformation [Botsch and Kobbelt 2004], Poisson deformation [Yu et al. 2004], Laplacian editing [Sorkine et al. 2004] and other linear variational surface deformation approaches (see also the survey in [Botsch and Sorkine 2008]). Volumetric information and rigidity are also incorporated to enhance the shape-preservation in [Igarashi et al. 2005; Botsch et al. 2006; Botsch et al. 2007; Sorkine and Alexa 2007]. One common drawback of these approaches is that the positions of vertices on a model need to be determined by solving a system of linear equations after every update of handles, which becomes a bottleneck of computation. A recent development in [Jacobson et al. 2011; Jacobson and Sorkine 2011; Jacobson et al. 2012b] transfers the workload from online optimization to offline. Specifically, the weights corresponding to handles are computed on every vertex of a model before manipulating the handles (similar to [Zayer et al. 2005]). The deformed shape is then evaluated by linear blending of transformations defined on handles. In [Sumner et al. 2007], the handles are elements of a simplified mesh. Although this strategy is more efficient than the deformation methods based on online optimization, they still cannot avoid solving large linear systems, which slows down the response of deformation after inserting new han-

dles. Moreover, the results of deformation are also suffered from the artificial distortions caused by the problems of meshes (e.g., too coarse meshes for a fine deformation, a mesh with ‘needle’ and ‘cap’ triangles, and the problem of symmetry). Our meshfree approach solves these problems by providing closed-form formulas to generate weights preserving all the demanded properties for producing deformations with high quality in real-time.

Another thread of researches for deformation focuses on mesh-independent approaches. Different handles are employed for shape manipulation. Points are used in [Yoshizawa et al. 2002; Schaefer et al. 2006], and curves are employed as handles in [Lazarus et al. 1994; Singh and Fiume 1998]. Grid-based deformation techniques in [Sederberg and Parry 1986; Lee et al. 1995] conduct the bivariate/trivariate cubic splines to realize deformations with C^2 -continuity. Users are allowed to move control points of the spline surfaces/solids to modify the embedded shapes, where the editing is indirect. Some approaches have been developed to extend this approach to provide the ability of direct editing (ref. [Hsu et al. 1992; Hu et al. 2001]). However, the computational domain is still limited to a simple topology (i.e., genus zero). An improvement of the grid-based techniques is introduced by Beier and Neely [1992] to allow handles in the form of line segments by using the Shepard’s interpolation [Shepard 1968]. Cage-based deformation (e.g., [Joshi et al. 2007; Ben-Chen et al. 2009]) can be considered as a further generalization of grid-based deformation, where weights can be found by a closed-form in terms of the handles in [Ju et al. 2005; Lipman et al. 2008]. However, the construction of cages is usually not automatic and the manipulation on cages instead of a model itself is indirect.

Moving least square (MLS) strategy is employed in [Schaefer et al. 2006] for interpolating the similarity/rigid deformation at handle points. A closed-form solution is provided in their approach to determine the transformation matrix on every point in a MLS manner. The transformations in the whole domain need to be computed when any handle is moved. In other words, the deformation is globally affected by all handles – lack of sparsity. Different from this MLS approach, our approach belongs to the category of linear blending based deformation. When the property of sparsity is preserved on the weights, the deformation at a point is only affected by the nearby handles that is easier to be predicted by end-users. Moreover, the deformation determined by our approach is resolution independent, which is very important for image manipulation.

The work of generating weights for linear blending also relates to the research of scattered data interpolation, where *radial basis functions* (RBF) are widely used (e.g., [Floater and Iske 1996; Botsch and Kobbelt 2005]). In [Botsch and Kobbelt 2005], the deformation is governed by global RBFs that lead to a dense linear system to be solved. The weights determined by the dense (or global) data interpolation approaches lack of sparsity. Therefore, every point in the domain is changed when any handle is updated even if it is far away. Although the *compactly supported radial basis functions* (CSRBF) can help on introducing the sparsity (ref. [Floater and Iske 1996]), it does not provide closed-form formulas as our approach.

3 Meshfree Weighting

Following the linear blending formulation, the new position of a point $\mathbf{p} \in \Omega$ is determined by the transformations \mathbf{T}_i defined on handles H_i as¹

$$\mathbf{p}' = \sum_{i=1}^m w_i(\mathbf{p}) \mathbf{T}_i \mathbf{p} \quad (1)$$

¹ \mathbf{T}_i is a homogenous matrix and \mathbf{p} is represented by homogeneous coordinate.

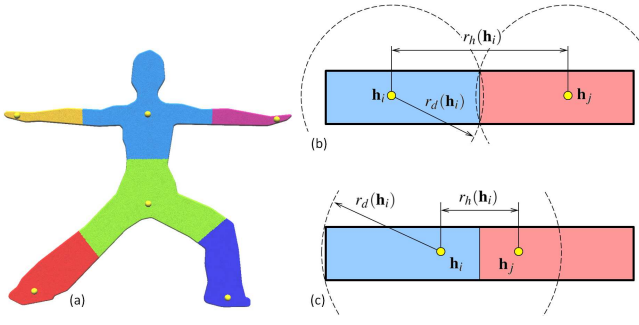


Figure 2: Voronoi diagram based method to determine the size of local support. (a) The Voronoi diagram of handles can decompose Ω into smaller pieces. (b) The illustration of $r_h(\mathbf{h}_i)$ and $r_d(\mathbf{h}_i)$ in the Voronoi diagram. (c) Very close handles can lead to $r_h(\mathbf{h}_i) < r_d(\mathbf{h}_i)$.

with $w_i(\cdot)$ being the scalar field of weights to be determined. The origin of a handle H_i is denoted by \mathbf{h}_i . This linear blending based deformation is fast and easy-to-implement. However, carelessly assigned weights can lead to visible artifacts in results. Basically, a deformation with high quality must have the following properties:

- **Smoothness:** The scalar field of weights must be smooth to avoid visual artifact (discontinuity) in both 2D and 3D deformations. We use compactly supported Bézier basis functions in our formulation, which lead to a weight field with C^2 -continuity.
- **Interpolation:** The final transformation determined by the linear blending must interpolate the transformations at the handles. Specifically, the weight on a handle H_i is *one* at its origin while basis functions centered at other handles give *zero* at this point. This is guaranteed by the locality and the sparsity in our formulation.
- **Consistency:** When applying the same transformation \mathbf{T} on all handles, all points in Ω must be consistently transformed by \mathbf{T} . This is enforced by the partition-of-unity property in our formulation. Another consistency requirement is about direction. The region influenced by a handle should not change in the inverse direction of the transformation assigned on the handle. We ensure this by the property of non-negativity.
- **Shape-awareness:** This is a property more or less subjective. Basically, the intrinsic requirement on shape-awareness is to have deformations like stretching, bending and twisting an elastic solid, where the handles serve as pins. In our formulation, this is preserved by 1) having non-positive first derivative of basis functions and 2) letting all basis functions have similar support sizes. No-local-maxima on weights will prevent generating singularity (e.g., a point moves faster than all its neighbors) during deformation.

Our formulation below leads to C^2 -continuous weights preserving all these properties in deformations.

3.1 Formulation

Each handle H_i is equipped with a compactly supported basis function with support size r_i as $\phi_i(d(\mathbf{p}, \mathbf{h}_i)/r_i)$, where \mathbf{h}_i is the location of H_i and $d(\cdot, \cdot)$ returns the intrinsic-distance (see Appendix A for the definition) between two points inside Ω . The scalar field of

the weights for H_i is then defined as

$$w_i(\mathbf{p}) = \frac{\phi_i(d(\mathbf{p}, \mathbf{h}_i)/r_i)}{\sum_{j=1}^m \phi_j(d(\mathbf{p}, \mathbf{h}_j)/r_j)}, \quad (2)$$

which enforces the partition-of-unity.

To be shape-aware and interpolate handles, $\phi_i(\cdot)$ is chosen as a monotonically decreasing function with $\phi_i(0) = 1$ and $\phi_i(t) = 0$ ($\forall t \geq 1$). A quintic polynomial is employed for the function $\phi_i(t)$ so that the constraints for C^1 and C^2 -continuity at the boundary of the supporting regions can be satisfied. Specifically, we need

$$\phi_i'(0) = \phi_i'(1) = \phi_i''(0) = \phi_i''(1) = 0. \quad (3)$$

To ease the evaluation and analysis, each $\phi_i(t)$ is represented as the y -component (i.e., $\phi_i(t) = \mathbf{b}^y(t)$, $t \in [0, 1]$) of a 2D Bézier curve with degree- n ($n \geq 5$)

$$\mathbf{b}(t) = \sum_{i=0}^n \mathbf{b}_i B_{i,n}(t), \quad (4)$$

where $B_{i,n}(t)$ are the Bernstein polynomials. From the property of Bézier curves (ref. [Farin 2002]), we know that $x = t$ when $\mathbf{b}_i^x = i/n$. Letting $\mathbf{b}_{0,1,2}^y \equiv 1$ and $\mathbf{b}_{n,n-1,n-2}^y \equiv 0$ can satisfy these constraints at the endpoints (see Appendix B for more details). For the rest control points, we can simply assign them as 0.5 or align them along the line $\mathbf{b}_2 \mathbf{b}_{n-2}$ uniformly.

When the intrinsic-distance is used to generate the input parameter t for the basis functions, linear blending based deformations driven by these basis functions behave in a shape-aware manner. Now the problem left is how to determine the support size r_i of each basis function. As a basic requirement of handle-driven deformation based on linear blending, every point $\mathbf{p} \in \Omega$ should be influenced by at least one handle. To be shape-aware, a point \mathbf{p} should be mostly affected by its closest handle in Ω . Voronoi diagram sited at the origins of handles $\{\mathbf{h}_i\}$ provides an intrinsic decomposition of Ω according to these observations (see Fig.2(a)), where the intrinsic-distance in Ω is used as the metric for generating the Voronoi diagram. We denote the cell that corresponds to \mathbf{h}_i by $\mathcal{V}(\mathbf{h}_i)$. Two metrics according to a handle H_i can be defined as follows (see Fig.2(b) for an illustration):

- The size of a Voronoi cell: $r_d(\mathbf{h}_i) = \sup_{\mathbf{q} \in \mathcal{V}(\mathbf{h}_i)} d(\mathbf{q}, \mathbf{h}_i)$;
- The separation to other sites: $r_h(\mathbf{h}_i) = \inf_{\mathbf{h}_j (j \neq i)} d(\mathbf{h}_i, \mathbf{h}_j)$.

To let the basis function $\phi_i(t)$ centered at H_i cover all points in $\mathcal{V}(\mathbf{h}_i)$ and to ensure the handle interpolation property, it should have

$$r_d(\mathbf{h}_i) < r_i \leq r_h(\mathbf{h}_i). \quad (5)$$

The support size can be $r_i = (1 - \alpha)r_d(\mathbf{h}_i) + \alpha r_h(\mathbf{h}_i)$ with $\alpha \in (0, 1]$ being specified by users as a shape factor. For most of the examples in this paper, $\alpha = 1$ is used. It is possible to have two handles too close to each other so that $r_h(\mathbf{h}_i) < r_d(\mathbf{h}_i)$ (see Fig.2(c) for an example). For solving such cases, we will use the virtual handle insertion algorithm (presented in Section 4).

3.2 Analysis and Discussion

We analyze the advantages of our formulation for the handle-driven deformation based on linear blending.

Non-negativity: $\phi_i(t) \geq 0$ so that $\forall \mathbf{p} \in \Omega, w_i(\mathbf{p}) \geq 0$. Moreover, when $r_i > r_d(\mathbf{h}_i)$ is ensured for all handles, every point in Ω should be covered by at least one handle's support. In other words, $\sum_{j=1}^m \phi_j(\cdot) \neq 0$.

Partition-of-unity: This has been enforced by the formulation in Eq.(2). That is,

$$\sum_{i=1}^m w_i(\mathbf{p}) = \sum_{i=1}^m \frac{\phi_i(d(\mathbf{p}, \mathbf{h}_i)/r_i)}{\sum_{j=1}^m \phi_j(d(\mathbf{p}, \mathbf{h}_j)/r_j)} \equiv 1.$$

Locality/Sparsity: This is preserved by $\forall t \geq 1, \phi_i(t) \equiv 0$ and the condition given in Eq.(5). The transformation at a point coincident with a handle is only determined by the handle itself. $\forall i \neq j, \phi_j(\mathbf{h}_i) \equiv 0$.

Smoothness: C^2 -continuity is preserved on the weights determined by Eq.(2). First of all, the basis function $\phi_i(t) = \mathbf{b}^y(t)$ is C^n -continuous for $t \in (0, 1)$ when $\mathbf{b}^y(t)$ is defined as a Bézier curve in Eq.(4) with $n \geq 5$. Therefore, $w_i(\mathbf{p})$ is also C^n -continuous when $\phi_j(d(\mathbf{p}, \mathbf{h}_j)/r_j) \neq 0$ for any other $j \neq i$. In the region that is only covered by the support of H_i , $w_i \equiv 1$. Similarly, it is also a constant function ($w_i \equiv 0$) in the region outside the support of H_i . By Eq.(3), it is not difficult to prove the C^2 -continuity at the following two cases:

- i) $d(\mathbf{p}, \mathbf{h}_j) < r_j$ and $d(\mathbf{p}, \mathbf{h}_i) = r_i$,
- ii) $d(\mathbf{p}, \mathbf{h}_j) = r_j$ and $d(\mathbf{p}, \mathbf{h}_i) < r_i$,

where both the first and second derivatives are zero.

No-local-maxima: The global maxima of a weight w_i only happens at the origin of handle H_i and the regions only covered by the support of H_i . Besides, we also observe the phenomenon of no-local-maxima in all our experimental tests.

Closed-form: The weights $\{w_i(\mathbf{p})\}$ at any point $\mathbf{p} \in \Omega$ are evaluated in a closed-form (i.e., by Eq.(2)). This guarantees the flexibility of inserting new handles during the deformation in real-time.

Meshfree: As the evaluation of basis functions to determine the weights is only related to the intrinsic-distance from points to the origin of handles, the solution is independent of mesh quality and resolution. In the mesh-dependent solutions, elements with poor shape, which can occur after a drastic deformation step, must be optimized. Remeshing leads to another round of weights computation that could be time-consuming.

In short, our method preserves all the merits of prior methods for linear blending based deformation (e.g., [Jacobson et al. 2011; Jacobson and Sorkine 2011; Jacobson et al. 2012b]) while introducing new benefits of flexibility and efficiency.

Besides the flexibility of inserting new handles during the deformation, we also provide users a method to change the behavior of handles by adjusting the shape of basis functions (i.e., $\phi_i(t)$). For example, as shown in Fig.3, for the basis function $\phi_i(t)$ built by a septic Bézier curve ($n = 7$), we can assign different values to \mathbf{b}_3^x and \mathbf{b}_4^x to obtain different shapes for $\phi_i(t)$ to have different deformation behaviors. Basically, a ‘flat’ basis function (e.g., $\mathbf{b}_3^x = \mathbf{b}_4^x = 0.5$) results in a deformation simulating hard materials while a more curved basis function (e.g., $\mathbf{b}_3^x = 1, \mathbf{b}_4^x = 0$) makes the deformation soft. When using polynomials in higher orders, we have more degree-of-freedom to change the shape of basis function. However, according to our experiments, septic polynomials are good enough in most of the cases.

The formulation of meshfree weighting also has some limitations. First, the interpolation property cannot be preserved when the distance between two handles are too close while the regions to be covered by either handle are large. Specifically, the interpolation of handles becomes an approximation when $r_d(\mathbf{h}_i) < r_h(\mathbf{h}_i)$ in Eq.(5) can NOT be satisfied. Second, for the region that is only covered by one handle, the transformation is consistent with the handle. Then, the deformation presented in this region is not shape-aware

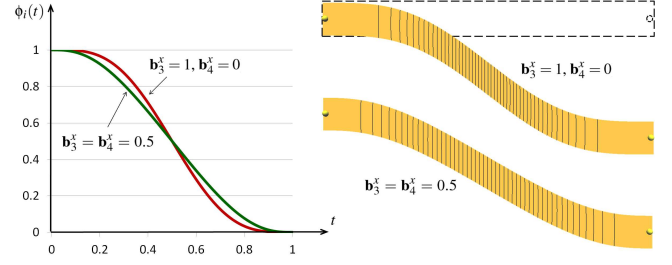


Figure 3: By using different basis functions in our formulation, different distributions of weights can be obtained which lead to the change of deformation behaviors. Isocurves for the weight field of the right handle are also shown in black lines on the deformed bars.

— i.e., the influence of handles does not decay while increasing the distance to the handle’s center. Both the problems will be solved by applying the virtual handle insertion approach presented in the following section.

4 Virtual Handle Insertion

A handle insertion algorithm is developed to enrich our meshfree weighting framework in the aspects of guaranteeing the handle interpolation property and improving the shape-awareness of deformation.

4.1 Insertion algorithm

When $r_d(\mathbf{h}_i) > r_h(\mathbf{h}_i)$, we know that there are points in the voronoi cell $\mathcal{V}(\mathbf{h}_i)$ whose distances to \mathbf{h}_i are larger than the minimal distance from \mathbf{h}_i to other handles.

Proposition 1 When $r_d(\mathbf{h}_i) > r_h(\mathbf{h}_i)$, inserting new sites at the points $\mathbf{h}_d \in \mathcal{V}(\mathbf{h}_i)$ with $d(\mathbf{h}_d, \mathbf{h}_i) = r_d(\mathbf{h}_i)$ can reduce $r_d(\mathbf{h}_i)$ while keeping $r_h(\mathbf{h}_i)$ unchanged.

Proof. First of all, the value of $r_h(\mathbf{h}_i)$ is not affected. When \mathbf{h}_d is the only point in $\mathcal{V}(\mathbf{h}_i)$ with $d(\mathbf{h}_d, \mathbf{h}_i) = r_d(\mathbf{h}_i)$, it is obvious $\exists \mathbf{q} \in \mathcal{V}(\mathbf{h}_i)$ with $d(\mathbf{q}, \mathbf{h}_d) < d(\mathbf{q}, \mathbf{h}_i)$. Define $\mathcal{S}(\mathbf{h}_d) = \{\mathbf{q} \in \mathcal{V}(\mathbf{h}_i) \mid d(\mathbf{q}, \mathbf{h}_d) < d(\mathbf{q}, \mathbf{h}_i)\}$. After inserting a new site at \mathbf{h}_d , the points in $\mathcal{S}(\mathbf{h}_d)$ become the member of $\mathcal{V}(\mathbf{h}_d)$. When all points with $d(\mathbf{h}_d, \mathbf{h}_i) = r_d(\mathbf{h}_i)$ have been assigned to other voronoi cells, the value of $r_d(\mathbf{h}_i)$ reduces. On the other aspect, the distances from the newly inserted points to \mathbf{h}_i are $r_d(\mathbf{h}_i)$ which is greater than $r_h(\mathbf{h}_i)$. \square

Based on this proposition, we develop a greedy algorithm for handle insertion. Define \mathcal{H} as the set of handles and $\delta(\cdot) = r_d(\cdot) - r_h(\cdot)$. When $\exists \mathbf{h}_i \in \mathcal{H}$ with $\delta(\mathbf{h}_i) > 0$, new handles are inserted to resolve this problem by reducing $\max_{\mathbf{h}_i \in \mathcal{H}} \{\delta(\mathbf{h}_i)/r_d(\mathbf{h}_i)\}$. The pseudo-code is described as **Algorithm Virtual Handle Insertion**.

Remarks. From Proposition 1, we know that inserting new handles in a voronoi cell $\mathcal{V}(\mathbf{h}_i)$ with $\delta(\mathbf{h}_i) > 0$ can reduce the value of $\delta(\mathbf{h}_i)$. However, inserting a new site $\mathbf{h}_d \in \mathcal{V}(\mathbf{h}_i)$ can also affect the other handles (i.e., H_j with $j \neq i$). In extreme cases, the original $\delta(\mathbf{h}_j) < 0$ could be turned into $\delta(\mathbf{h}_j) > 0$. Then, new handles need to be added into $\mathcal{V}(\mathbf{h}_j)$.

Our virtual handle insertion algorithm can be considered as a variant of the farthest point sampling algorithm, which tends to tessellate a domain into a voronoi diagram with neighboring voronoi cells having similar sizes. The condition of $r_d(\cdot) < r_h(\cdot)$ is satisfied on

Algorithm 1: Virtual Handle Insertion

Input: the set \mathcal{H} of real handles

Output: the expanded set \mathcal{H} with virtual handles

while $\exists \mathbf{h}_i \in \mathcal{H}, \delta(\mathbf{h}_i) > 0$ **do**

 Find the handle $\mathbf{h}_m = \arg \max_{\mathbf{h}_i \in \mathcal{H}} \delta(\mathbf{h}_i)/r_d(\mathbf{h}_i)$;

 Find a point $\mathbf{p} \in \mathcal{V}(\mathbf{h}_m)$ with $d(\mathbf{p}, \mathbf{h}_m) = r_d(\mathbf{h}_m)$;

 Insert a new handle located at \mathbf{p} into \mathcal{H} ;

 Update the values of $r_d(\cdot)$ and $r_h(\cdot)$ on all handles;

end

return \mathcal{H} ;

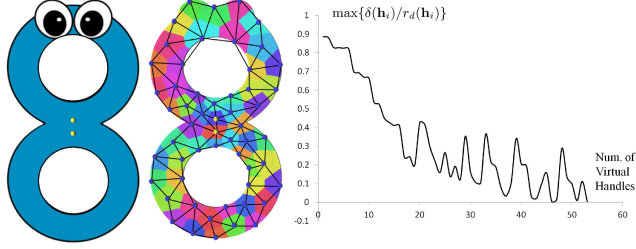


Figure 4: When two handles are too close to each other (see left), the condition for interpolation (i.e., $r_d(\cdot) < r_h(\cdot)$) can only be satisfied after inserting virtual handles. (Middle) The newly inserted virtual handles (in blue) tessellate the deformation domain into voronoi cells whose areas are similar to neighboring cells. The Delaunay graph, $\mathcal{DG}(\mathcal{H} \cup \mathcal{H}^v)$, of the voronoi diagram is also shown – see the network linking the handles. (Right) The score, $\max\{\delta(\mathbf{h}_i)/r_d(\mathbf{h}_i)\}$, of our Virtual Handle Insertion algorithm drops while inserting virtual handles.

all handles when this is the case. Our experimental tests also follow this observation (see Fig.4 for an example).

4.2 Transformation on Virtual Handles

A left problem is how to determine the transformation on virtual handles according to the user-specified transformations on real handles. Denote the set of real handles as \mathcal{H} and the set of virtual handles as \mathcal{H}^v . As aforementioned, the handles of $\mathcal{H} \cup \mathcal{H}^v$ have partitioned the given domain Ω into a voronoi diagram $\text{Vor}(\mathcal{H} \cup \mathcal{H}^v)$. A dual graph of $\text{Vor}(\mathcal{H} \cup \mathcal{H}^v)$ can be constructed by 1) using the sites of every voronoi cell as nodes and 2) linking the sites of every two neighboring voronoi cells by a straight line, which is a Delaunay graph [Berg et al. 2008]. We denote the Delaunay graph by $\mathcal{DG}(\mathcal{H} \cup \mathcal{H}^v)$ and also use symbol H to represent nodes in \mathcal{DG} since each node is in fact a handle (real or virtual). The transformations of handles in \mathcal{H}^v are determined with the help of the Delaunay graph as follows.

- For each handle H_i in \mathcal{H} , a harmonic field $\varpi_i(\cdot)$ is computed on \mathcal{DG} to assign each handle H_g a field value $\varpi_i(H_g)$. Boundary conditions, $\varpi_i(H_i) = 1$ and $\varpi_i(H_{j \neq i}) = 0$, are given to compute the harmonic field $\varpi_i(\cdot)$. If there are m handles in \mathcal{H} , m harmonic fields are determined on \mathcal{DG} .
- After converting the transformation \mathbf{T}_i of each handle into a rotation quaternion \mathbf{q}_i and a translation vector \mathbf{t}_i , the rotation and the translation on a virtual handle $H_v \in \mathcal{H}^v$ can be determined by

$$\begin{pmatrix} \mathbf{q}_v \\ \mathbf{t}_v \end{pmatrix} = \frac{1}{\varpi_{\text{sum}}(H_v)} \sum_{H_i \in \mathcal{H}} \varpi_i(H_v) \begin{pmatrix} \mathbf{q}_i \\ \mathbf{t}_i \end{pmatrix} \quad (6)$$

with $\varpi_{\text{sum}}(\cdot) = \sum_{H_j \in \mathcal{H}} \varpi_j(\cdot)$.

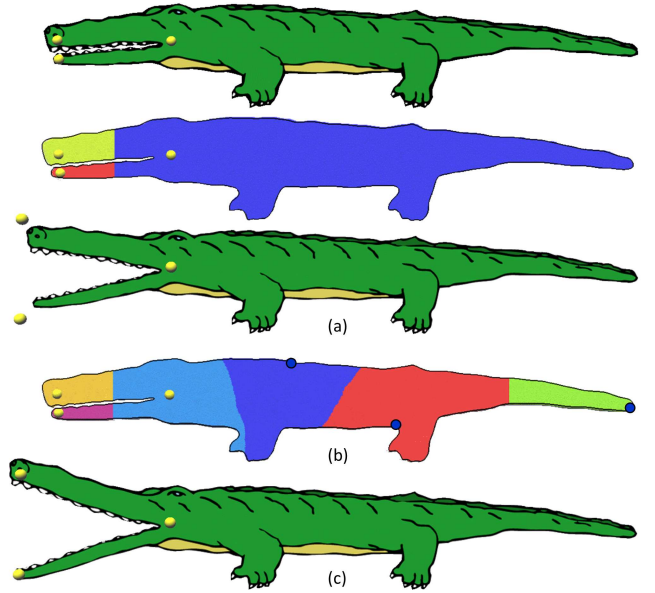


Figure 5: A handle covering a large region can affect the interpolation on its nearby handles. (a) For the handle at the right, its voronoi cell covers all the right part of the alligator – this leads to a value of $r_d(\cdot)$ that is much larger than $r_h(\cdot)$. In this case, transformations at the left two handles cannot be interpolated. (b) Virtual handles (in blue color) are added to resolve the problem by the insertion algorithm. As a result, the domain to be deformed has been decomposed into smaller voronoi cells with handles (real and virtual) as sites. (c) The deformation result is driven by both the real and the virtual handles, where the transformations at real handles are interpolated.

- Finally, the quaternion and the translation determined on each virtual handle are converted back into a transformation matrix to be used in linear blending.

The transformation of virtual handles determined in this way brings in the effect of shape-awareness during the deformation. As illustrated in Figs.5 and 6, the deformation of whole domain driven by the transformations on handles (real and virtual) is very natural. The influence of a real handle decays when the distance to it increases.

5 Implementation Details

Similar to many other meshfree approaches, we sample the input domain Ω to be deformed into a set of dense points \mathcal{P} . By searching k -nearest-neighbors of each point, a graph $\mathcal{G}(\mathcal{P})$ spanning Ω (in discrete form) can be established by using points in \mathcal{P} as nodes and adding links between neighboring points. Note that user specified handles should also be added into \mathcal{P} to construct the graph (i.e., $\mathcal{H} \subset \mathcal{P}$). The intrinsic-distance from any point $\mathbf{q} \in \mathcal{P}$ to a handle is approximated by the distance between \mathbf{q} and the handle on the graph, which can be computed efficiently with the help of Dijkstra’s algorithm. Also, the voronoi diagram $\text{Vol}(\mathcal{H})$ can be obtained by the Dijkstra’s algorithm with multiple sources on $\mathcal{G}(\mathcal{P})$, where each sample is assigned to a voronoi cell. As the primitives used in the computation are points, the deformation approach can be easily generalized from 2D images to 3D solids. More examples can be found in the following section. To determine the weights on a general point $\mathbf{p} \in \Omega$ that is not a sample in \mathcal{P} , a linear blending based on reciprocal distance weights [Floater and Reimers 2001] is

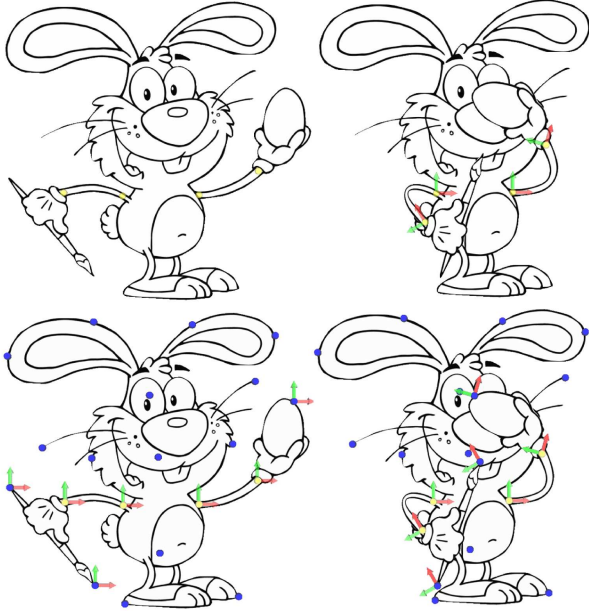


Figure 6: The deformation of a rabbit is drive by four real handles (see the yellow dots and the frames shown in the top row). The result of deformation is determined with the help of virtual handles (shown in blue dots). The transformations at handles (both real and virtual ones) are illustrated by frames.

employed to obtain the weight on \mathbf{p} from its k -nearest-neighbors in \mathcal{P} . There are more sophisticated parameterization strategies in [Floater and Reimers 2001], which can also be applied here. With the help of this meshless parameterization, we can easily take an up-sampling step in the domain Ω when the point set \mathcal{P} becomes sparse when applying a drastic deformation.

After using the virtual handle insertion algorithm to generate a set of new handles, harmonic fields are computed on a dual graph of $Vol(\mathcal{H})$ to determine the transformations on virtual handles. By our boundary condition, all field values are non-negative when uniform Laplacian is employed [Wardetzky et al. 2007]. In other words, the coefficients used in Eq.(6) are non-negative. Instead of solving a linear system to compute the harmonic field, we initially assign the field values on all real handles as one and the weights on all virtual handles are set as zero. Then we apply Laplacian operators to update their field values iteratively. The field values on virtual handles can be efficiently obtained after tens of iterations.

The point handles can be generalized to different types of handles (e.g., line segments and polygons, etc.). Specifically, each handle H_g now becomes a set of points $\{\mathbf{h}_g\}$ instead of a single point while all these points are equipped with the same transformation \mathbf{T}_g . The major change is the method to evaluate the intrinsic-distance from a query point \mathbf{q} to handles (e.g., line segments), which is the intrinsic-distance to \mathbf{q} 's closest sample point on the handle. The rest of our approach will keep unchanged. Extreme case occurs when two line-segment handles have a common endpoint so that $r_h(\cdot)$ of these two handles becomes zero. There is no way to satisfy the condition of $r_d(\cdot) < r_h(\cdot)$ for handle interpolation. We therefore only approximate the transformations specified on handles. Specifically, the basis function is changed to a global Gaussian

$$\phi_i(t) = e^{-(c_i t)^2} \quad (7)$$

with c_i being a constant to control the width of Gaussian. In our

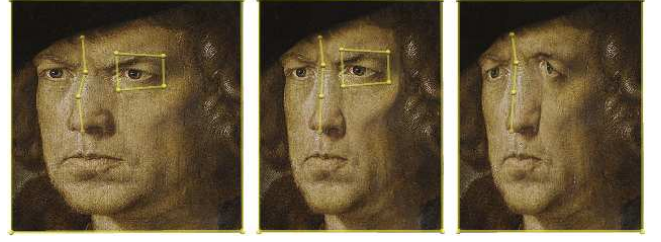


Figure 7: A portrait is edited by segment handles. Salient feature inside the closed loop of segment handles at the left eye is preserved after the deformation.



Figure 8: An example of processing the photograph of Tibet palace by segment handles. The image can be easily warped with the help of segment handles provided in our framework.

implementation, letting c_i be $\frac{1}{2}r_h(\cdot)$ works well in all tests. As some handles may have common endpoints, $r_h(\cdot)$ is changed to the minimal *non-zero* distance to other handles to exclude those connected handles. It is clear that the transformation at the position of a handle H_i is commonly determined by all handles in \mathcal{H} although the influence of far away handles is trivial. On the other aspect, the smoothness of deformation is improved to C^∞ . Cages can be formed by linking the segment handles into closed loops. For example when editing the portrait shown in Fig.7, the cage located at the boundary help resize the image. Moreover, the cage at the left eye fully controls the shape inside it and therefore preserves the salient feature.

6 Results

Our meshfree weighting method provides a compact tool to assign continuous weights for all points in the domain of deformation. With the help of sophisticated techniques for assigning transformations on the handles (e.g., the pseudo-edge method in [Jacobson et al. 2011] or the optimization method in [Jacobson et al. 2012a]), a natural user interface for shape deformation can be achieved.

We have tested this approach in a variety of examples by using both the point and the segment handles. Figures 1, 5 and 6 have already demonstrated the functionality of point handles. Especially, in Fig.5, the scheme of virtual handles insertion guarantees the interpolation at real handles. Figure 6 illustrates the effectiveness of our method in determining transformations on virtual handles. The

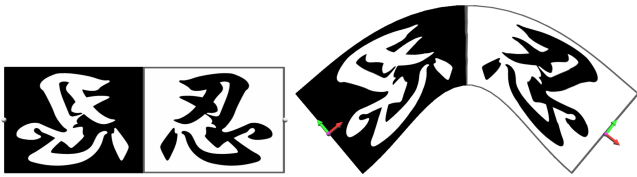


Figure 9: An example of symmetric deformation: when applying symmetric transformations on two symmetric handles to deform a symmetric domain, our meshfree approach guarantees to obtain a symmetric result.

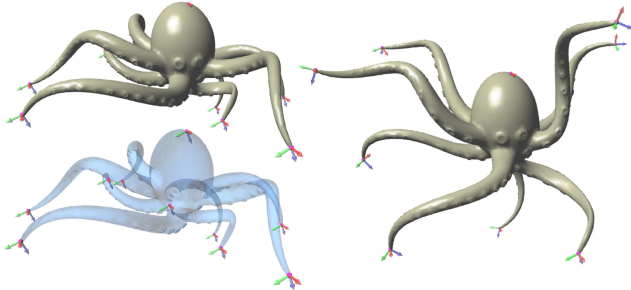


Figure 10: The flexible 3D Octopus can be easily manipulated by using the point handles.

example of using segment handles to deform a portrait has been shown in Fig.7. Another example is given in Fig.8 to warp the shape of palace. To obtain natural bending results, we can add rotations on handles by heuristic methods (e.g., the pseudo-edge [Jacobson et al. 2011]). Another example is to demonstrate the performance of our approach in a symmetric deformation. When deforming a symmetric domain by adding symmetric transformations on symmetric handles, it is expected to get a symmetric result. This property is preserved by our formulation (see Fig.9).

We also apply this method to deform 3D models. In these examples, the 3D models are represented by polygonal mesh surfaces. The weights computed by our approach are used in a linear blending way to determine the new positions of vertices. Note that, the space enclosed by a mesh surface need to be sampled into points with the help of voxelization technique (e.g., [Schwarz and Seidel 2010]) in order to evaluate the discrete intrinsic-distance in the domain to be deformed. Point handles are used to manipulate the flexible Octopus in Fig.10, where the interface of manipulation becomes user-friendly after employing the scheme of pseudo-edges to determine the transformation of point handles. Linear blending scheme is widely employed in the animation of skeletal models (e.g., [Jacobson and Sorkine 2011; Magnenat-Thalmann et al. 1988]). The example shown in Fig.11 gives the performance of our approach in this scenario. 3D models with very complex topology (e.g., the Buddha model with internal truss structures in Fig.12) that are hard to be meshed can be easily handled in our approach. When deformations with large rotation are applied (e.g., in Fig.13), a progressive deformation strategy can help generate satisfactory results.

For prior mesh-based approaches, the numerical system must be solved once more when new handles are inserted. In our mesh-free weighting formulation, the time cost of adding new handles is very trivial as the weights are determined in a closed-form. Table 1 lists the statistics of our approach on different examples. All the tests are conducted on a computer with Intel Core i7-3740QM CPU at 2.70GHz with 8GB memory, where our current implementation only uses a single-core. All results of deformation can be obtained

	$ \mathcal{H} $	$ \mathcal{S} $	t_{Vol} (sec.)	t_w (sec.)
Gingerman	2 (8)	155,457	0.584	0.054
Alligator	3 (3)	53,225	0.128	0.015
Rabbit	4 (15)	22,972	0.128	0.015
Portrait	11	4,225	0.029	0.005
Palace	22	4,225	0.041	0.003
Chinese	2	4,076	0.008	0.001
Octopus	10	7,485	0.039	0.002
Armadillo	17	26,002	0.100	0.014
Buddha	4	236,661	0.302	0.051
	20	236,661	0.834	0.160
Bar	2	4,765	0.009	0.002

Table 1: Computational Statistics for the examples shown in the paper. $|\mathcal{H}|$ denotes the number of handles (the number of virtual handles is shown in the bracket) and $|\mathcal{S}|$ represents the number of sample points used in the computation. The columns under t_{Vol} and t_w state the time used in the computation of the voronoi diagram and the weights respectively.

at an interactive speed.

Discussion. When using the meshfree formulation presented in the paper to deform real 2D/3D objects, sample points are adopted as the medium for realizing the computation. The error-bound of computation on this discrete representation is guaranteed by the density of samples. However, during the process of a sequence of deformations, the density of points could be changed dramatically. In this sense, a dynamic up-sampling step should be integrated in the framework to preserve the error-bound of intrinsic-distance computation. The image editing applications can be implemented by using either the super-sampling technique or the texture mapping on a mesh. In our framework, the cost of weight evaluation is trivial after resampling. The bottleneck is the computation of intrinsic-distances on the sample points. Our current implementation is based on the Dijkstra’s algorithm. However, this shortest path problem with multiple sources can be computed in parallel on the system with many-cores [Rong et al. 2011], which can result in a significant speedup and will be implemented in our future work.

Our formulation gives global maximum at the positions of handles, which is very important to avoid the unintuitive behavior of deformations. For a shape-aware deformation, it is also demanded having no-local-maximum. This has been verified in our experimental tests. We check the topology of isocurves on the fields of weights (see Fig.14 for an example). If there is a closed loop formed by isocurves of $w_i(\cdot)$ at one place except the center of the handle $h_i(\cdot)$, a local maximum is generated there. However, no such case is found in all our examples.

7 Conclusion

We present a method to determine weights of blending for shape deformation. Our formulation is meshfree and in a closed-form, which can be easily used in a variety of applications in 2D/3D deformations. Equipped with a virtual handle insertion algorithm, good properties of weights generated by prior mesh-based methods can all be preserved in this approach. A variety of examples have been shown to demonstrate the function of our approach.

Only linear blending deformations are tested in the paper. We plan to further extend the application of weights generated in this approach to more advanced skinning methods, such as dual quaternion [Kavan et al. 2008], with which the blending of two rigid motions will result in a rigid motion. This is a very important property when the deformation of articulated characters is computed by

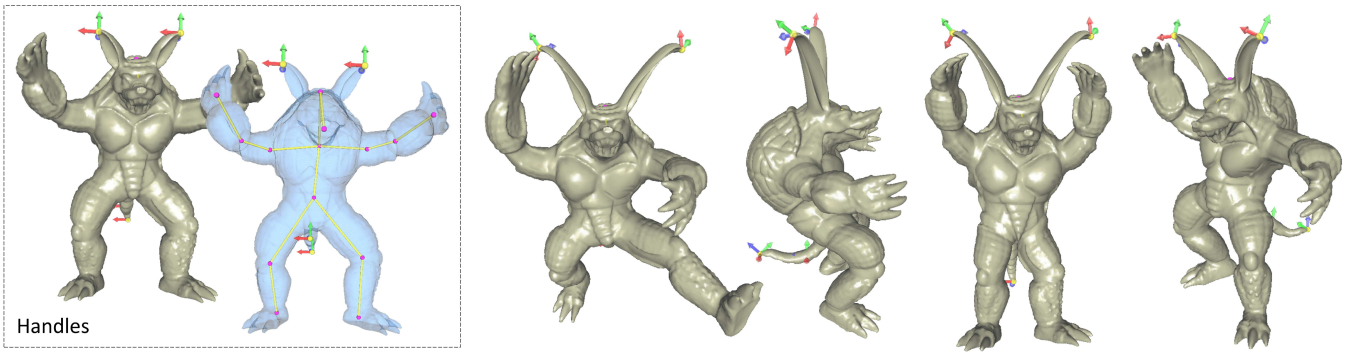


Figure 11: An example of using the weights determined by our approach in the animation of Armadillo driven by the point and the segment handles.

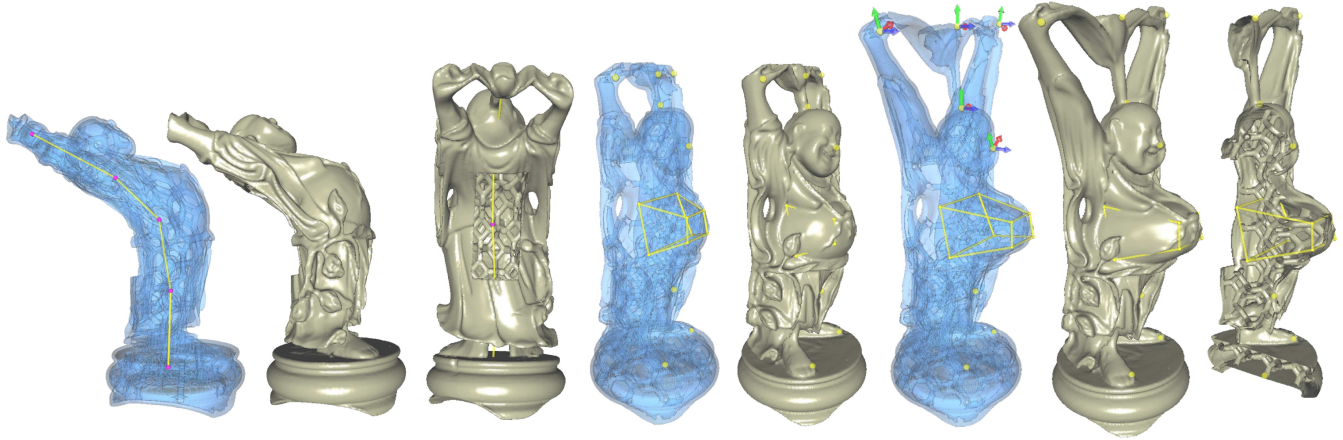


Figure 12: An example of deforming a model with very complex topology – the Buddha model with interior truss structures, where our meshfree approach can determine the weights for linear blending effectively and efficiently.

the skinning methods. The deformations driven by linear blending are not always injective and therefore can generate the results with foldovers and self-intersection. Recently, some researches have been conducted in this direction to produce injective mappings (e.g., [Aigerman and Lipman 2013; Schüller et al. 2013]), which are mainly mesh-based. In a function based formulation, the injectivity of a mapping can be checked by the sign of Jacobian. However, it is still not clear about how to resolve the problem when self-intersection is detected. This will be one of our future work.

References

- AIGERMAN, N., AND LIPMAN, Y. 2013. Injective and bounded distortion mappings in 3D. *ACM Trans. Graph.* 32, 4, 106:1–106:14.
- BEIER, T., AND NEELY, S. 1992. Feature-based image metamorphosis. *SIGGRAPH Comput. Graph.* 26, 2 (July), 35–42.
- BEN-CHEN, M., WEBER, O., AND GOTSMAN, C. 2009. Variational harmonic maps for space deformation. *ACM Trans. Graph.* 28, 3 (July), 34:1–34:11.
- BERG, M. D., CHEONG, O., KREVELD, M. V., AND OVERMARS, M. 2008. *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer-Verlag TELOS, Santa Clara, CA, USA.

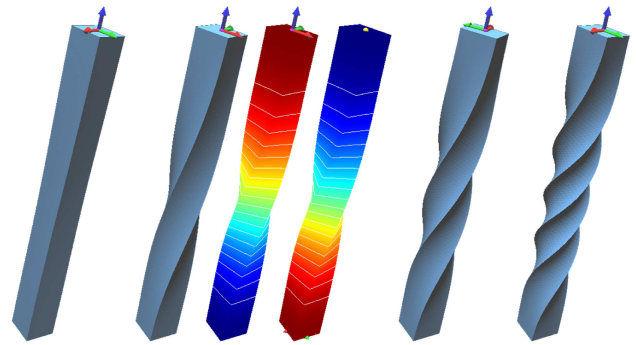


Figure 13: An example of progressively twisting a bar with sharp edges in different rotations: $\pi/4$, $\pi/2$ and 2π . The color maps show the distribution of weights according to two handles. The twists with large rotations are generated by progressively applying small rotations – e.g., 2° per update in our practice.

BOTSCH, M., AND KOBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.* 23, 3 (Aug.), 630–634.

BOTSCH, M., AND KOBELT, L. 2005. Real-time shape editing using radial basis functions. *Comput. Graph. Forum* 24, 3, 611–

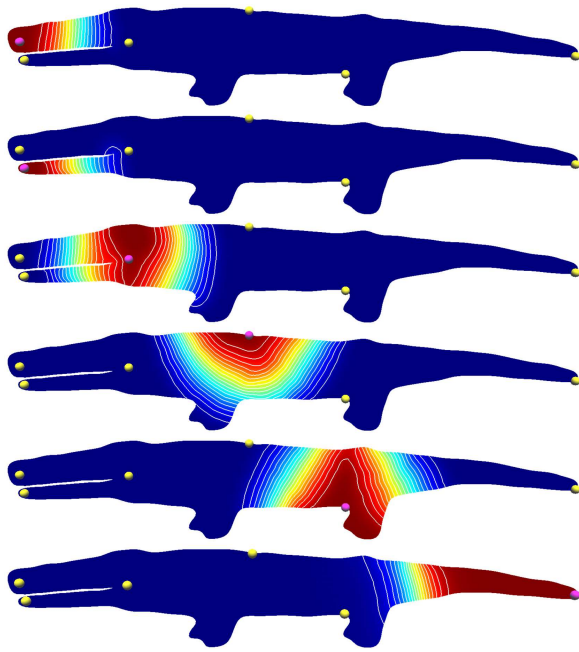


Figure 14: The verification of no-local-maximum is taken by analyzing the topology of isocurves on the weights' scalar-fields. The handles (real and virtual) in this example are the ones shown in Fig.5.

621.

- BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics* 14, 1 (Jan.), 213–230.
- BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, Eurographics Association, SGP '06, 11–20.
- BOTSCH, M., PAULY, M., WICKE, M., AND GROSS, M. H. 2007. Adaptive space deformations based on rigid cells. *Comput. Graph. Forum* 26, 3, 339–347.
- FARIN, G. 2002. *Curves and Surfaces for CAD: A Practical Guide*, 5th ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- FLOATER, M. S., AND ISKE, A. 1996. Multistep scattered data interpolation using compactly supported radial basis functions. *Journal of Computational and Applied Mathematics* 73, 1-2, 65–78.
- FLOATER, M. S., AND REIMERS, M. 2001. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design* 18, 2, 77–92.
- HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. *SIGGRAPH Comput. Graph.* 26, 2 (July), 177–184.
- HU, S.-M., ZHANG, H., TAI, C.-L., AND SUN, J.-G. 2001. Direct manipulation of ffd: efficient explicit solutions and decomposable multiple point constraints. *The Visual Computer* 17, 6, 370–379.
- IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.* 24, 3 (July), 1134–1141.
- JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph.* 30, 6 (Dec.), 165:1–165:8.
- JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4 (July), 78:1–78:8.
- JACOBSON, A., BARAN, I., KAVAN, L., POPOVIĆ, J., AND SORKINE, O. 2012. Fast automatic skinning transformations. *ACM Trans. Graph.* 31, 4, 77:1–77:10.
- JACOBSON, A., WEINKAUF, T., AND SORKINE, O. 2012. Smooth shape-aware functions with controlled extrema. *Comp. Graph. Forum* 31, 5 (Aug.), 1577–1586.
- JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph.* 26, 3 (July).
- JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.* 24, 3 (July), 561–566.
- KAVAN, L., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.* 27, 4, 105:1–105:23.
- LAZARUS, F., COQUILLART, S., AND JANCÉNE, P. 1994. Axial deformations: an intuitive deformation technique. *Computer-Aided Design* 26, 8, 607–613.
- LEE, S.-Y., CHWA, K.-Y., AND SHIN, S. Y. 1995. Image metamorphosis using snakes and free-form deformations. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ACM, SIGGRAPH '95, 439–448.
- LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. *ACM Trans. Graph.* 27, 3 (Aug.), 78:1–78:10.
- MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88*, 26–33.
- MILLIRON, T., JENSEN, R. J., BARZEL, R., AND FINKELSTEIN, A. 2002. A framework for geometric warps and deformations. *ACM Trans. Graph.* 21, 1 (Jan.), 20–51.
- RONG, G., LIU, Y., WANG, W., YIN, X., GU, X. D., AND GUO, X. 2011. Gpu-assisted computation of centroidal voronoi tessellation. *IEEE Transactions on Visualization and Computer Graphics* 17, 3, 345–356.
- SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Trans. Graph.* 25, 3 (July), 533–540.
- SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)* 32, 5, 125–135.
- SCHWARZ, M., AND SEIDEL, H.-P. 2010. Fast parallel surface and solid voxelization on gpus. *ACM Trans. Graph.* 29, 6, 179:1–179:10.

- SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.* 20, 4 (Aug.), 151–160.
- SHEPARD, D. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM, 517–524.
- SINGH, K., AND FIUME, E. 1998. Wires: A geometric deformation technique. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, SIGGRAPH '98, 405–414.
- SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, Eurographics Association, SGP '07, 109–116.
- SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, ACM, SGP '04, 175–184.
- SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph.* 26, 3 (July).
- VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2006. Vector field based shape deformations. *ACM Trans. Graph.* 25, 3 (July), 1118–1125.
- WARDETZKY, M., MATHUR, S., KÄLBERER, F., AND GRINSPUN, E. 2007. Discrete laplace operators: No free lunch. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, 33–37.
- YOSHIZAWA, S., BELYAEV, A., AND SEIDEL, H. P. 2002. A simple approach to interactive free-form shape deformations. In *Proceedings of 10th Pacific Conference on Computer Graphics and Applications*, 471–474.
- YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.* 23, 3 (Aug.), 644–651.
- ZAYER, R., RÖSSL, C., KARNI, Z., AND SEIDEL, H.-P. 2005. Harmonic guidance for surface deformation. *Comput. Graph. Forum* 24, 3, 601–609.

Appendix A: Intrinsic-Distance

For a 2-manifold shape in 2D/3D Euclidean space, all points on the shape form a bounded domain Ω . For any two points $\{\mathbf{p}_s, \mathbf{p}_e\} \in \Omega$, if there exists a curve line $\mathcal{C} \subset \Omega$ connecting \mathbf{p}_s and \mathbf{p}_e , we define the intrinsic-distance of $\{\mathbf{p}_s, \mathbf{p}_e\}$ along the curve \mathcal{C} as

$$d(\mathbf{p}_s, \mathbf{p}_e; \mathcal{C}) = \text{length}(\mathcal{C})$$

Then the intrinsic-distance of $\{\mathbf{p}_s, \mathbf{p}_e\}$ in Ω is defined as

$$d(\mathbf{p}_s, \mathbf{p}_e) = \min_{\mathcal{C}} \text{length}(\mathcal{C}).$$

If there is no curve connecting \mathbf{p}_s and \mathbf{p}_e , that is the case they are not located in a connected region of Ω . The intrinsic-distance is then defined as $d(\mathbf{p}_s, \mathbf{p}_e) = \infty$.

Sampling based intrinsic-distance. For a set of sampling points $\mathcal{S} \in \Omega$ of Ω , we can build a graph \mathcal{G} by using the sample points as nodes. We represent the shortest distance between \mathbf{p}_s and \mathbf{p}_e on \mathcal{G}

as $d_{\mathcal{G}}(\mathbf{p}_s, \mathbf{p}_e; \mathcal{S})$. If for any two points $\{\mathbf{p}_s, \mathbf{p}_e\} \in \Omega$, we always have

$$|d_{\mathcal{G}}(\mathbf{p}_s, \mathbf{p}_e; \mathcal{S}) - d(\mathbf{p}_s, \mathbf{p}_e)| \leq \varepsilon,$$

the sampling \mathcal{S} is a *distance-bounded sampling* of Ω .

The intrinsic-distance defined in this way has the following properties:

- **Existence:** $d(\mathbf{p}_s, \mathbf{p}_e; \mathcal{C})$ is always calculable once \mathcal{C} is determined, which is a curve segment in Ω . Therefore, $d(\mathbf{p}_s, \mathbf{p}_e)$ always exists for Ω when \mathbf{p}_s and \mathbf{p}_e are located in the same connected region.
- **Uniqueness:** $d(\mathbf{p}_s, \mathbf{p}_e)$ is uniquely determined while the corresponding curves may be multiple.
- **Convergency:** For any $\varepsilon > 0$, there always exists an infinite sampling of Ω – that is the sampling density $D(\mathcal{S}) \rightarrow \infty$. Since $\lim_{D(\mathcal{S}) \rightarrow \infty} \varepsilon = 0$, we have

$$\lim_{D(\mathcal{S}) \rightarrow \infty} |d_{\mathcal{G}}(\mathbf{p}_s, \mathbf{p}_e; \mathcal{S}) - d(\mathbf{p}_s, \mathbf{p}_e)| = 0.$$

Appendix B: Endpoint Constraints

From the analysis in [Farin 2002], we know that

$$\mathbf{b}'(0) = n(\mathbf{b}_1 - \mathbf{b}_0), \quad \mathbf{b}'(1) = n(\mathbf{b}_n - \mathbf{b}_{n-1})$$

for a Bézier curve in n -th order. And also

$$\mathbf{b}''(0) = n(n-1)(\mathbf{b}_2 - 2\mathbf{b}_1 + \mathbf{b}_0)$$

$$\mathbf{b}''(1) = n(n-1)(\mathbf{b}_n - 2\mathbf{b}_{n-1} + \mathbf{b}_{n-2})$$

Incorporating the constraints in Eq.(3), we have

$$\mathbf{b}_1 = \mathbf{b}_0, \mathbf{b}_n = \mathbf{b}_{n-1}, \mathbf{b}_1 = \frac{\mathbf{b}_0 + \mathbf{b}_2}{2}, \mathbf{b}_{n-1} = \frac{\mathbf{b}_n + \mathbf{b}_{n-2}}{2}.$$

As we already need $\mathbf{b}_i^x = i/n$ to let $x = t$, it is not difficult to find that $\mathbf{b}_0^y = \mathbf{b}_1^y = \mathbf{b}_2^y = 1$ and $\mathbf{b}_n^y = \mathbf{b}_{n-1}^y = \mathbf{b}_{n-2}^y = 0$ satisfy all these constraints.